BÀI TẬP LÀM THÊM CHƯƠNG 3

**Bài 02:**

**Phương pháp Insertion Sort**

40 70 20 60 90 10 50 30

Giả sử phần tử đầu tiên đã có thứ tự

i=1, xét phần tử a[1] > a[0] => chèn a[1] vào bên phải a[0]

40 70 20 60 90 10 50 30

i=2, xét phần tử a[2] < a[0] => chèn a[2] vào bên trái a[0]

20 40 70 60 90 10 50 30

i=3, xét phần tử a[3] > a[0] => chèn a[3] vào bên phải a[0]

20 40 70 60 90 10 50 30

i=4, xét phần tử a[3] < a[4] => chèn a[3] vào bên trái a[4]

20 40 70 60 90 10 50 30

i=5, xét phần tử a[5] < a[0] => chèn a[5] vào bên trái a[0]

10 20 40 70 60 90 50 30

i=6, xét phần tử a[6] > a[2] => chèn a[6] vào bên trái a[2]

10 20 40 50 70 60 90 30

Mảng sau khi sắp tăng dần: 10 20 30 40 50 60 70 90

**Phương pháp SelectionSort**

40 70 20 60 90 10 50 30

Bước 1: Xét vị trí từ 0 đến n-1 => min\_pos=5

Swap(0, min\_pos)

10 70 20 60 90 40 50 30

i++;

Bước 2: Xét vị trí từ 1 đến n-1 => min\_pos = 2

Swap(1,min\_pos)

10 20 70 60 90 40 50 30

i++;

Bước 3: Xét vị trí từ 2 đến n-1 => min\_pos = 7

Swap(2,min\_pos)

10 20 30 60 90 40 50 70

i++;

Bước 4: Xét vị trí từ 3 đến n-1 => min\_pos = 5

Swap(3,min\_pos)

10 20 30 40 90 60 50 70

Bước 5: Xét vị trí từ 4 đến n-1 => min\_pos = 6

Swap(4,min\_pos)

10 20 30 40 50 60 90 70

Bước 6: Xét vị trí từ 5 đến n-1 => min\_pos = 5

Không swap

10 20 30 40 50 60 90 70

Bước 7: Xét vị trí từ 6 đến n-1 => min\_pos = 7

Swap(6,min\_pos)

10 20 30 40 50 60 70 90

Mảng sau khi sắp tăng dần: 10 20 30 40 50 60 70 90

**Phương pháp InterchangceSort**

20 30 4 5 15

**Bước 1**: i=0, j=i+1=1;

Xét a[0] < a[1] => swap(a[0], a[1]);

30 20 4 5 15

Xét a[0] > a[2] =>không swap

Xét a[0] > a[3] => không swap

Xét a[0] > a[n-1] => stop

30 20 4 5 15

**Bước 2**: i=1, j=2;

Xét a[1] > a[2] => không swap

Xét a[1] > a[3] => không swap

Xét a[1] > a[n-1] => stop

30 20 4 5 15

**Bước 3**: i=2, j=3;

Xét a[2] < a[5] => swap (a[2], a[5]);

30 20 5 4 15

Xét a[2] < a[4] => swap(a[2], a[4]);

30 20 15 4 5

Xét a[2] > a[n-1] => stop

30 20 15 4 5

**Bước 4**: i=3, j=4;

Xét a[3] < a[4] => swap(a[3], a[4]);

30 20 15 5 4

Xét a[3] > a[n-1] => stop

Mảng sau khi sắp xếp giảm dần: 30 20 15 5 4

Phương pháp BubbleSort:

19 22 10 15 99 11

**Bước 1**: i=0;

Xét a[5] < a[4] => Không swap

Xét a[4] > a[3] => Swap(a[4], a[3]);

19 22 10 99 15 11

Xét a[3] > a[2] => Swap(a[3], a[2]);

19 22 99 10 15 11

Xét a[2] > a[1] => Swap(a[2], a[1]);

19 99 22 10 15 11

Xét a[1] > a[0] => Swap(a[1], a[0]);

99 19 22 10 15 11

j=i=0 => stop

i++;

**Bước 2**: i=1;

99 19 22 10 15 11

Xét a[5] < a[4] => Không swap

Xét a[4] > a[3] => Swap(a[4], a[3]);

99 19 22 15 10 11

Xét a[3] < a[2] => Không swap

Xét a[2] > a[1] => Swap(a[2], a[1]);

99 22 19 15 10 11

j=i=1 => stop

i++;

**Bước 3**: i=2;

99 22 19 15 10 11

Xét a[5] >a[4] => Swap (a[5], a[4]);

99 22 19 15 11 10

Xét a[4] <a[3] => Không swap

Xét a[3] <a[2] => Không swap

j=i=2 => stop

i++;

**Bước 4:** i=3;

99 22 19 15 11 10

Xét a[5] < a[4] => Không swap

Xét a[4] < a[3] => Không swap

j=i=3 => stop

i++;

**Bước 5**: i=4;

99 22 19 15 11 10

Xét a[5] < a[4] => Không swap

j=i=4 =>stop

Mảng sau khi sắp xếp giảm dần: 99 22 19 15 11 10

**Phương pháp QuickSort:**

86 32 23 19 9 27 12 10

**Bước 1**: left=0, right=7, x=a[(left+right)/2]=a[3]=19

i=0, a[0]=86 > x=19 => i++

i=1, a[1]=32 > x=19 => i++

i=2, a[2]=23 > x=19 => i++

i=3, a[3]=19 < x=19 => dừng

j=7, a[7]=10 < x=19 => j--

j=6, a[6]=12 < x=19 => j--

j=5, a[5]=27 > x=19 => dừng

swap(a[3], a[5]);

86 32 23 27 9 19 12 10

i++, j--;

**Bước 2**: i=j=a[4]

Xét (left <j) Đệ qui Quick sort(a, left, j) ⬄ QuickSort(a,0,4)

Xét (i < right) Đệ qui Quick sort(a, right, i) ⬄ QuickSort(a, 7,4)

**Bước 3**: QuickSort(a, 0,4)

86 32 23 27 9

left=0, right=4, x=a[2]=23

i=0 a[0]=86 > x=23 => i++

i=1 a[1]=32 > x=23 => i++

i=2 a[2]=23 < x=23 => dừng

j=4 a[4]=9 < x=23 => j--

j=3 a[3]=27 > x=23 => dừng

swap(a[3], a[2]);

86 32 27 23 9

i++, j-- ⬄ i=3, j=2

do( j < i ) => đệ qui

Xét (left < j) QuickSort(a,0,2)

Xét (i < right) QuickSort(a,4,3)

**Bước 4**: QuickSort(a,4,7)

9 19 12 10

Left=4, right=7, x=a[5]=19;

i=4 xét a[4]=9 < a=19 => dừng

j=7 a[7]=10 < x=19 => j--

j=6 a[6]=12 < x=19 => j--

j=5 a[5]=19 < x=19 => dừng

swap(a[4], a[5]);

19 9 12 10

i++, j-- ⬄ i=5, j=4;

do(j < i) => đệ qui

Xét ( left < j) => Không thể xảy ra

Xét (i < right) QuickSort(a, i, right) ⬄ QuickSort(a,5,7)

**Bước 5**: QuickSort(a,5,7)

9 12 10

left=5, right=7, x=a[6]=12;

i=5 xét a[5]=9 < x=12 =>dừng

j=7 xét a[7]=10 < x=12 => j--

j=6 xét a[6]=12 < x=12 => dừng

swap(a[5], a[6]);

12 9 10

i++, j-- ⬄ i=6, j=5

do(j < i) => đệ qui

Xét (left < j) => không thể xảy ra

Xét (i < right) QuickSort(a,i,right) ⬄ QuickSort(a,6,7)

**Bước 6**: QuickSort(a,6,7)

9 10

left=6, right=7, x=a[6]=9;

i=6 xét a[6]=9 < x=9 => dừng

j=7 xét a[7]=10 > x=9 => dừng

swap(a[6], a[7]);

10 9

i=n-1 => dừng đệ qui

Mảng sau khi sắp giảm dần: 86 32 27 23 19 12 10 9

**Phương pháp HeapSort:**

86 32 23 19 9 27 12 10

Dãy đầu: a[0] -> a[3]

Dãy sau: a[4] -> a[7]

**Bước 1**:

i=3, so sánh a[3], a[7] => min= a[7]=10. Swap(a[3], a[7]);

86 32 23 10 9 27 12 19

i--;

i=2, so sánh a[2], a[5], a[6] => min= a[6]=12. Swap(a[2], a[6]);

86 32 12 10 9 27 23 19

i--;

i=1, so sánh a[1], a[3], a[4] => min= a[4]=9. Swap(a[1], a[4]);

86 9 12 10 32 27 23 19

i--;

i=0, so sánh a[0], a[1], a[2] => min= a[1]=9. Swap(a[0], a[1]);

9 86 12 10 32 27 23 19

Xét tính lan truyền tại a[1], a[3], a[4] => min= a[3]=10. Swap(a[1], a[3]);

9 10 12 86 32 27 23 19

Xét tính lan truyền tại a[3], a[7]=> min= a[7]=19. Swap(a[7], a[3]);

9 10 12 19 32 27 23 86

Việc tạo heap(heap min) hoàn tất: Swap(a[0], a[7])

86 10 12 19 32 27 23 **9**

**Bước 2**:

i=0, so sánh a[0], a[1], a[2] => min= a[1]=10. Swap(a[0], a[1]);

10 86 12 19 32 27 23 **9**

Xét tính lan truyền tại a[1], a[3], a[4]=> min= a[3]=19. Swap(a[1], a[3]);

10 19 12 86 32 27 23 **9**

Việc tạo heap(heap min) hoàn tất: Swap(a[0], a[6])

23 19 12 86 32 27 **10** **9**

**Bước 3:**

i=0, so sánh a[0], a[1], a[2] => min= a[2]=12. Swap(a[0], a[2]);

12 19 23 86 32 27 **10** **9**

Xét tính lan truyền tại a[2], a[5] => min= a[2]=23. Không lan truyền.

Việc tạo heap(heap min) hoàn tất: Swap(a[0], a[5])

27 19 23 86 32 12 **10** **9**

**Bước 4:**

i=0, so sánh a[0], a[1], a[2] => min= a[1]=19. Swap(a[0], a[1]);

19 27 23 86 32 **12** **10** **9**

Xét tính lan truyền tại a[1], a[3], a[4] => min= a[1]=27. Không lan truyền.

Việc tạo heap(heap min) hoàn tất: Swap(a[0], a[4])

32 27 23 86 19 **12** **10** **9**

**Bước 5:**

i=0, so sánh a[0], a[1], a[2] => min= a[2]=23. Swap(a[0], a[2]);

23 27 32 86 **19** **12** **10** **9**

Xét tính lan truyền tại a[2]. Không lan truyền.

Việc tạo heap(heap min) hoàn tất: Swap(a[0], a[3])

86 27 32 23 **19** **12** **10** **9**

**Bước 6:**

i=0, so sánh a[0], a[1], a[2] => min= a[1]=27. Swap(a[0], a[1]);

27 86 32 **23** **19** **12** **10** **9**

Xét tính lan truyền tại a[1]. Không lan truyền.

Việc tạo heap(heap min) hoàn tất: Swap(a[0], a[2])

32 86 27 **23** **19** **12** **10** **9**

**Bước 7**:

i=0, so sánh a[0], a[1] => min= a[0]=32. Không Swap(a[0], a[1]);

32 86 **27** **23** **19** **12** **10** **9**

Việc tạo heap(heap min) hoàn tất: Swap(a[0], a[1])

86 32 **27** **23** **19** **12** **10** **9**

**Danh sách được xếp thứ tự giảm dần:**

**86 32 27 23 19 12 10 9**

**2.7** Độ phức tạp của 90 trong thuật toán tìm kiếm tuần tự là: O(n)

Độ phức tạp của 90 trong thuật toán tìm kiếm nhị phân là: O(n2)